CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Digital Repository @ Iowa State University

IOWA STATE UNIVERSITY
Digital Repository

Center for Nondestructive Evaluation
Conference Papers, Posters and Presentations

Center for Nondestructive Evaluation

2018

Automated Construction and Insertion of Layer-by Layer Finite Element Sub-Models of Damaged Composites

Stephen D. Holland
Iowa State University, sdh4@iastate.edu

Adarsh Krishnamurthy
Iowa State University, adarsh@iastate.edu

Onur Bingol
Iowa State University, orbingol@iastate.edu

Robert Grandin
Iowa State University, rgrandin@iastate.edu

Follow this and additional works at: https://lib.dr.iastate.edu/cnde_conf

 Part of the Materials Science and Engineering Commons, and the Structures and Materials Commons

The complete bibliographic information for this item can be found at https://lib.dr.iastate.edu/cnde_conf/126. For information on how to cite this item, please visit http://lib.dr.iastate.edu/howtocite.html.

This Conference Proceeding is brought to you for free and open access by the Center for Nondestructive Evaluation at Iowa State University Digital Repository. It has been accepted for inclusion in Center for Nondestructive Evaluation Conference Papers, Posters and Presentations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

Automated Construction and Insertion of Layer-by Layer Finite Element Sub-Models of Damaged Composites

Abstract

Finite element models of composite structures are generally shell-based and modeled at the laminate level. More detailed layer-by-layer lamina-level models are sometimes needed for representing joints or for modeling defect growth processes. We describe a method and toolkit for automating the creation and insertion of layerby-layer finite element sub-models of composite laminate. We focus in particular on representing damage captured from nondestructive evaluation (NDE) measurements. The method is based on scripting existing simulation and solid modeling tools (ABAQUS and ACIS). It works even on complicated, curved CAD models. The submodel location is identified by the intersection of a cylinder with the structure. We then execute a series of instructions to generate a new shell with the submodel region removed, generate the layer-by-layer submodel, and bond together the layers and models with desired boundary conditions and defects. The instructions represent the steps of lamination and bonding for creating the composite. The output of the method includes CAD models of the new shell and each lamina within the submodel, and a Python script for ABAQUS that will load the CAD models, bond them together, and apply the specified boundary conditions.

Disciplines

Materials Science and Engineering | Structures and Materials

Comments

This proceeding appeared in Holland, Stephen D., Adarsh Krishnamurthy, Onur Bingol, and Robert Grandin. "Automated Construction and Insertion of Layer-by-Layer Finite Element Sub-Models of Damaged Composites." In *Proceedings of the American Society for Composites—Thirty-third Technical Conference (*2018). Lancaster, PA: DEStech Publications, Inc. DOI: 10.12783/asc33/26002. Posted with permission.

This conference proceeding is available at Iowa State University Digital Repository: https://lib.dr.iastate.edu/cnde_conf/126

Automated Construction and Insertion of Layer-by Layer Finite Element Sub-Models of Damaged Composites

STEPHEN D. HOLLAND, ADARSH KRISHNAMURTHY, ONUR BINGOL and ROBERT GRANDIN

ABSTRACT

Finite element models of composite structures are generally shell-based and modeled at the laminate level. More detailed layer-by-layer lamina-level models are sometimes needed for representing joints or for modeling defect growth processes. We describe a method and toolkit for automating the creation and insertion of layer-by-layer finite element sub-models of composite laminate. We focus in particular on representing damage captured from nondestructive evaluation (NDE) measurements. The method is based on scripting existing simulation and solid modeling tools (ABAQUS and ACIS). It works even on complicated, curved CAD models. The submodel location is identified by the intersection of a cylinder with the structure. We then execute a series of instructions to generate a new shell with the submodel region removed, generate the layer-by-layer submodel, and bond together the layers and models with desired boundary conditions and defects. The instructions represent the steps of lamination and bonding for creating the composite. The output of the method includes CAD models of the new shell and each lamina within the submodel, and a Python script for ABAQUS that will load the CAD models, bond them together, and apply the specified boundary conditions.

INTRODUCTION

Composite laminates can easily end up with damage, whether from manufacturing flaws, in-service impacts, or long-term degradation. Nondestructive evaluation (NDE) can be used to assess the presence of such damage, but in many cases it is difficult to determine whether a repair is warranted. Composite repairs are expensive and do not necessarily achieve the full original strength. If the damage does not af-

Holland et al., Iowa State University Center for NDE 1915 Scholl Road, Ames, IA 50011USA

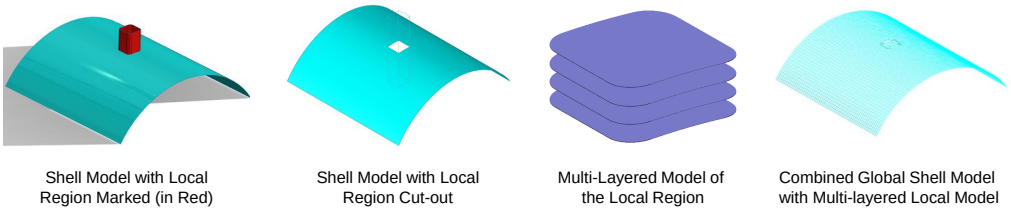


Figure 1. Overview of the model generation process. The local region is marked using a periodic (closed) surface (red). The intersection of the marked surface and the shell model is used to locate the local region. The layer-by-layer model is constructed in the local region and tied to the global shell model for analysis.

fect structural integrity or fatigue resistance, repair might not be necessary. To assess the need for repair or replacement, a model of the structural integrity of the damaged joint will be needed.

Finite element analysis is a primary tool used for evaluating the strength and stiffness of composite structures, substructures, joints, and even simple laminates. Finite element models of composite structures or major substructures are usually based on stiffened shells and modeled as laminates.

More detailed layer-by-layer lamina-level models are sometimes needed for representing joints, understanding failure processes, or for modeling defect growth processes. For example when damage or internal defects are a concern (especially when considering a fatigue scenario as opposed to ultimate strength) it is necessary to model the defect growth process in detail. The structural performance of a damaged region in terms of failure strength and delamination growth may depend on the orientations of individual plies adjacent to the damage, so a ply-by-ply model is needed in order to evaluate the effect of a defect on structural integrity. This paper describes a method and toolkit ("Delamo") for easily generating such layer-by-layer finite element submodels.

The usual manual process of creating layer-by-layer models is extremely tedious. The layers must be created so that they so they fit together perfectly. In curved geometries, this means using computer aided design (CAD) tools and offsetting operations. Any pre-existing damage to be simulated – such as a delaminated region or fiber breakage complicates matters further by the need to split the layer or its surfaces into multiple regions according to the damage.

The process of manually applying boundary conditions to the layer-by-layer model in the finite element (FE) software is even worse than the construction of the layers: Every pair of adjacent surfaces of adjacent laminae must be identified and selected, then an appropriate boundary condition (continuity, contact, or cohesive) with appropriate parameters must be configured. If the lamina level model represents a sub-region of a larger structure (typically modeled with shell elements), a precise matching hole needs to be cut in the larger structure. Then each layer can be bonded into place. Because this process has so many steps, each of which need to be exactly correct, it is extremely error-prone.

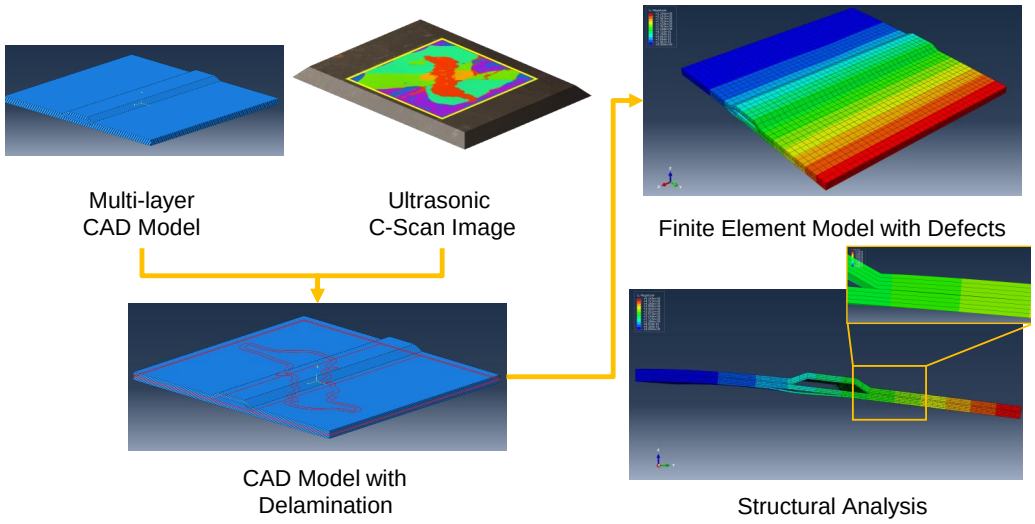


Figure 2. Steps in the creation of the layer-by-layer local model with defects and performing structural analysis.

Creation of layer-by-layer submodels can be made significantly more practical through scripting: Most CAD tools and FE packages support scripting of one form or another. A CAD script can be written to define the layers, and a FE script can be written to bond them together. Such a process is reasonably practicable for simpler geometries, such as where the layers are flat and not broken into subregions e.g. by fiber breakage or delamination. In these cases it is reasonably practical to identify surfaces for applying boundary conditions by simple coordinates.

For more complicated geometries with curved surfaces, damage, partial layers, stiffeners, etc. the required scripts become significantly more intricate, complicated, and error-prone themselves. They are difficult because what is naturally a single process – creating and bonding the layers – is forced into two separate pieces, the layer creation and the application of boundary conditions, without a natural mechanism for sharing information between those pieces. Worse, the scripts end up specific to the particular problem being simulated and are not readily reused for different problems.

Our method and toolkit, introduced in the previous year's proceedings [1], facilitate the creation of layer-by-layer finite element submodels as a single multi-step process. Our method is based on representing the construction and assembly of the submodel as a script with step-by-step instructions for creating and bonding layers. The modeling process is designed to follow as closely as possible the actual steps involved in manufacturing a composite laminate; layer creation and bonding can be interspersed as is natural. (see Figure 2) The script can also include instructions for bonding the sub-model into a larger model. It should be possible to model almost any laminar composite with such a script.

Our toolkit, known as "Delamo" [2] provides the means to execute these scripts including creation of layer-by-layer solid models and generating instructions to the finite element package to apply the desired boundary conditions. The current im-

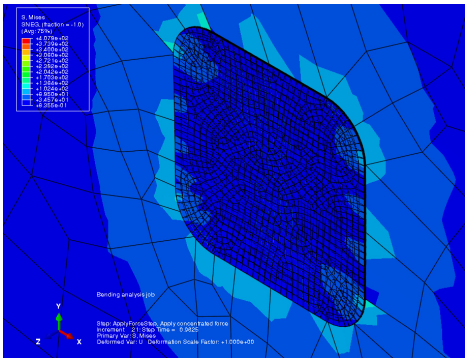


Figure 3. Example ABAQUS output from structural analysis applied to an interior layer-by-layer model surrounded by an outer shell

plementation is built on existing simulation and solid modeling tools (ABAQUS and ACIS). It works even on complicated, curved CAD models. Given an existing shell model of the structure, the submodel location can be identified by the intersection of a cylinder with the shell. Following the user-provided script we can generate a new shell with the submodel region removed, generate the layer-by-layer submodel, and bond together the layers and models with desired boundary conditions and defects.

The instructions in the script represent as primitive operations the cutting of the submodel region, the creation of layers, and the bonding of adjacent layers to create the composite. The output of the method includes CAD models of the new shell and each lamina within the submodel, and a Python script for ABAQUS [3] that will load the CAD models, bond them together, and apply the specified boundary conditions. Figure 3 shows example ABAQUS output from a finite element structural analysis applied to an outer shell model and an interior layer-by-layer model.

The method can generate a very wide range of possible laminates because the construction is specified by the same steps that would be used in manufacturing; if the laminate can be constructed by starting with a mold and then laying and bonding a series of layers, then it can be generated by our method. The steps are specified as Python function calls that operate on loaded CAD shells, lamina, or ABAQUS objects. Executing the steps directly uses ACIS [4] to perform solid modeling operations on the CAD shells and lamina, such as cutting out the submodel region, creating a mold, creating a solid layer from the mold, or creating a solid layer atop a prior layer. The steps can also include ABAQUS operations such as lamina bonding, shell-to-solid coupling, and applying boundary conditions. They can also include creating imperfect or frangible bonds. For example, delaminations can be represented with an interior contact zone and an outer cohesive zone that allows the delamination to grow under load.

ABAQUS operations cannot execute during the model generation phase because ABAQUS is not yet running (only the solid modeler). Instead, these operations are captured and stored for inclusion in the generated ABAQUS script. The model generation phase outputs a solid model (CAD file) and a Python script for ABAQUS that loads the CAD file and executes the ABAQUS operations such as defining sections, bonding layers, etc.

Related work

DESICOS [5] is a European Union project to improve the accuracy of predictions of compressive buckling strength of composite rocket components, so that a less conservative knockdown factor can be used in design. As part of the project, the DESICOS members have developed and published ABAQUS Python scripts to generate shells with certain types of imperfections. The types of defects modeled are a wide range of mostly manufacturing defects that could affect buckling: Dimples, incorrect fiber orientation, cutouts, nonuniform thickness, perturbation loads, etc. They are modeled on a shell basis, as opposed to the ply-by-ply models of arbitrary delaminations or fiber breakage models generated by Delamo.

TexGen [6] is a tool for generating microstructural models of textiles, including woven fabrics and 3D weaves used in advanced composites. It can assemble a model from individual yarns, predict and represent the individual yarn paths, and create a meshed or voxelized representation of the yarns for export to various possible analysis tools, including ABAQUS.

METHODS

Extraction of submodel region

An abbreviated script representing the process of creating a layer-by-layer submodel is shown in Figure 4. Aside from various boilerplate instructions that are required, the first step in creating a layer-by-layer submodel is to extract the desired submodel region. The desired submodel region should have been marked by the user in the original CAD file by a cylinder that intersects with the shell model of the structure. The cylinder is identified by its color. Delamo provides a function `shell_and_cutout_from_shelltool()` to perform the extraction. This function uses the ACIS CAD modeling engine to read in a CAD file, identify the cylinder (the "tool") by its color, and perform an intersection operation between the tool and shell to determine the cut line. The function then creates and returns a copy of the shell with the cutout removed, a copy of the cutout, and identification of the edges that surround the cutout. The data structure for representing the coupling between shell and solid also needs to be created from the shell and cutout, using the method `shell_solid_coupling.from_shell_and_cutout()`.

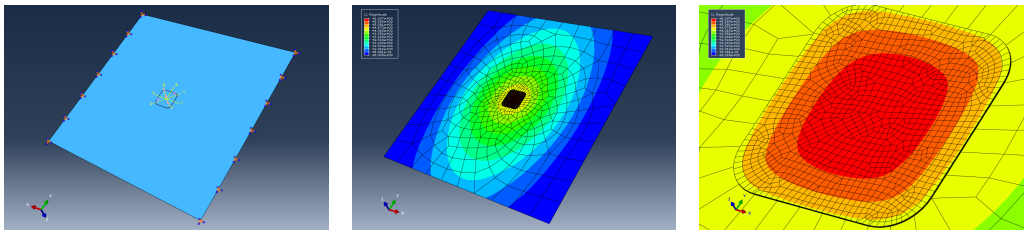
Creation of layer-by-layer finite element model

The ABAQUS finite element model needs to be properly initialized and parameters of the shell, such as material, section, and meshing parameters need to be defined. These are done using the functions and methods provided by ABAQUS or by simplified convenience wrappers around those functions.

The first layer is created using a CAD "offset" operation using the cutout as a mold, with the `Layer.CreateFromMold()` method. The layer needs to be finalized, meshed, and bonded to the surrounding shell with the

```
# Load shell and tool that defines cutout region
(shell,
 cutout,
 edgeinfo) = shell_and_cutout_from_shelltool(DM,"Shell+Tool.SAT")

# ABAQUS definition of material
ShellMat=FEModel.Material(name="ShellMat")
ShellMat.Elastic(table=((200e9/1.e6,0.22),))
# (similar definition of ABAQUS section omitted)

# Meshing of the shell
shell.MeshSimple(ShellMeshElemTypes,shellmeshsize,
                 abqC.QUAD_DOMINATED,abqC.SYSTEM_ASSIGN,
                 refinedmeshsize=12.0,refined_edges=edgeinfo,
                 pointtolerance=DM.pointtolerance,
                 tangenttolerance=DM.normaltolerance)

# Creation of shell-solid coupling object
ssc = shell_solid_coupling.from_shell_and_cutout(DM,shell,cutout)
for layernum in range(4):
  # Create layer
  if layernum==0:
    layer = Layer.CreateFromMold(DM,cutout.gk_part,
                                 delamo.CADwrap.OFFSET_DIRECTION,
                               thickness,"Layer %d" % (layernum),
                             LaminaSection,orientation[layernum])
  else:
    layer = Layer.CreateFromLayer(DM,prev_layer.gk_layer,
                                  delamo.CADwrap.OFFSET_DIRECTION,
                                thickness,"Layer %d" % (layernum),
                              LaminaSection,orientation[layernum])
  layer.Finalize(DM)   # Finalize components of layer

  layer.MeshSimple(MeshElemTypes,meshsize,abqC.HEX_DOMINATED,
                   abqC.SYSTEM_ASSIGN)

  layeroffset=thickness + (layernum+0.5)*thickness
  ssc.bond_layer(DM,layer,layeroffset) # Bond layer to shell

  if prev_layer is not None: # Bond layer to previous layer
    bond_layers(DM,prev_layer,layer,
                delamo.CADwrap.Delamination_COHESIVE,
                CohesiveInteraction,ContactInteraction)
  prev_layer=layer

ssc.Finalize(DM,influenceDistance=6.0)

# You would apply external ABAQUS boundary conditions here
DM.Finalize(cad_file_name,script_to_generate)
```

Figure 4. Example Delamo script for generating a submodel (some boilerplate code has been omitted for clarity).



(a) (b) (c)

Figure 5. Shell-solid coupling of the 5-layered planar composite laminate region with the shell model. Subfigure (a) shows the boundary conditions and loading setup of the shell model and the 5-layered composite local region constructed from the shell model, (b) shows the displacement field ($U_{magnitude}$) of the shell model meshed coarsely (3.0 mm), (c) shows the displacement field in the local region of the composite laminate.

`bond_layer()` method. Subsequent layers are created from the prior layer with the `Layer.CreateFromLayer()` method. Each of these layers needs likewise to be meshed, bonded to the surrounding shell, and finalized. Each also needs to be bonded to the prior layer with the `bond_layers()` function.

External loading and boundary conditions can be defined through ABAQUS Python functions and methods. Delamo supplies methods such as `GetInstanceEdgeRegion()` on the shell and layer objects to help identify edges and faces by location and (optionally) orientation.

Once all boundary conditions are defined, the shell-to-solid coupling object and the finite element model as a whole need to be finalized, generating an output CAD file with the shell (minus cutout), the cutout, and the layers, as well as an ABAQUS Python script that will load in the output CAD file and assemble the finite element model.

Automated insertion of damage

If you know a-priori where damage should be it can be inserted during the initial creation, for example by adding additional parameters to `bond_layers()` indicating a delamination outline. Automated insertion of damage from NDE data is more complicated because of the need to determine the exact location of the damage.

The first step in automated insertion of damage is to run the model generation script with no damage, with loops unrolled[1], and with `bond_layers()` instructions uniquely identified. Each layer bonding instruction will also generate a geometric representation of the bonding surface, represented as a triangular mesh STL file.

Delamination damage detected in the NDE signals can then be mapped onto the nearest bonding surface. Delamination outlines are represented as a loops of coordinates on those surfaces. Since the loops of coordinates are referenced to a bond-

[1]Loop unrolling is a computer science term of art referring to the process of replacing loops in the instruction sequence with repetitive copies of the instructions. It is needed to separate out iterative `bond_layers()` instructions so that they can later have damage data attached.

ing surface, they can be attached via an additional parameter to the corresponding `bond_layers()` instruction. Executing the sequence of instructions will create a new CAD model and FE script that include the measured damage.

CONCLUSIONS

Our method and toolkit helps bridge the gap between mechanics-oriented studies on progressive damage parameters such as interfacial fracture toughness and the need for structural analysis of complicated real-world geometries. It makes it practical to insert disbonds and other defects into a structural model and utilize the exact same simulation tool and models (ABAQUS) employed in the mechanic studies. It works by representing the model creation process as a series of solid modeling and finite element modeling steps. It is general because the steps required to create a layer-by-layer submodel follow the steps used to manufacture a physical part. Our toolkit generates a CAD file with the needed solid models and a Python script for ABAQUS that loads and configures the simulation. The process of creating a solid layer-by-layer lamina submodel of a defect region within a larger shell simulation is radically simplified compared to prior alternatives.

ACKNOWLEDGMENTS

This work is supported by National Aeronautics and Space Administration (NASA) under contract number NNL15AA12C through the Center for Non-Destructive Evaluation (CNDE) at Iowa State University and is a part of the NASA Advanced Composites Project. We would like to thank Dr. William P. Winfree and Dr. Cheryl A. Rose of NASA for their valuable input and support for this project. We would also like to thank Dr. Ronald Roberts from Iowa State University CNDE for ultrasonic scan data of actual composite delaminations and Dr. Ashraf Bastawros of Iowa State University for his assistance with ABAQUS cohesive modeling.

REFERENCES

1. Holland, S. D., Krishnamurthy, A., Bingol, O., and Grandin, R., Automated Construction of Layer-by-Layer Finite Element Sub-Models of Damaged Composites Based on NDE Data, In *Proc. Amer. Soc. Composites 32nd Tech. Conf* 2017. `http://dx.doi.org/10.12783/asc2017/15216`

2. O.R. Bingol, B. Schiefelbein, R.J. Grandin, S.D. Holland, and A. Krishnamurthy, An Integrated Framework for Solid Modeling and Structural Analysis of Layered Composites with Defects, Under Review, CAD (2018)

3. Dassault Systemes, ABAQUS `http://www.3ds.com/products-services/simulia/products/abaqus/` Accessed June 16, 2017.

4. Dassault Systemes, 3D ACIS Modeler https://www.spatial.com/products/3d-acis-modeling Accessed June 16, 2017.

5. DESICOS: New Robust Design Guidline for Imperfection SensitiveComposite Launcher Structures http://www.desicos.eu, Last viewed June 16, 2017.

6. Lin, H., L. P. Brown, and A. C. Long. 2011. "Modeling and Simulating Textile Structures using TexGen," *Advanced Materials Research* 331:44-47.